

Direct Addressing + Open Addressing

Note Title

10/4/2007

Data Structures of Set ADT

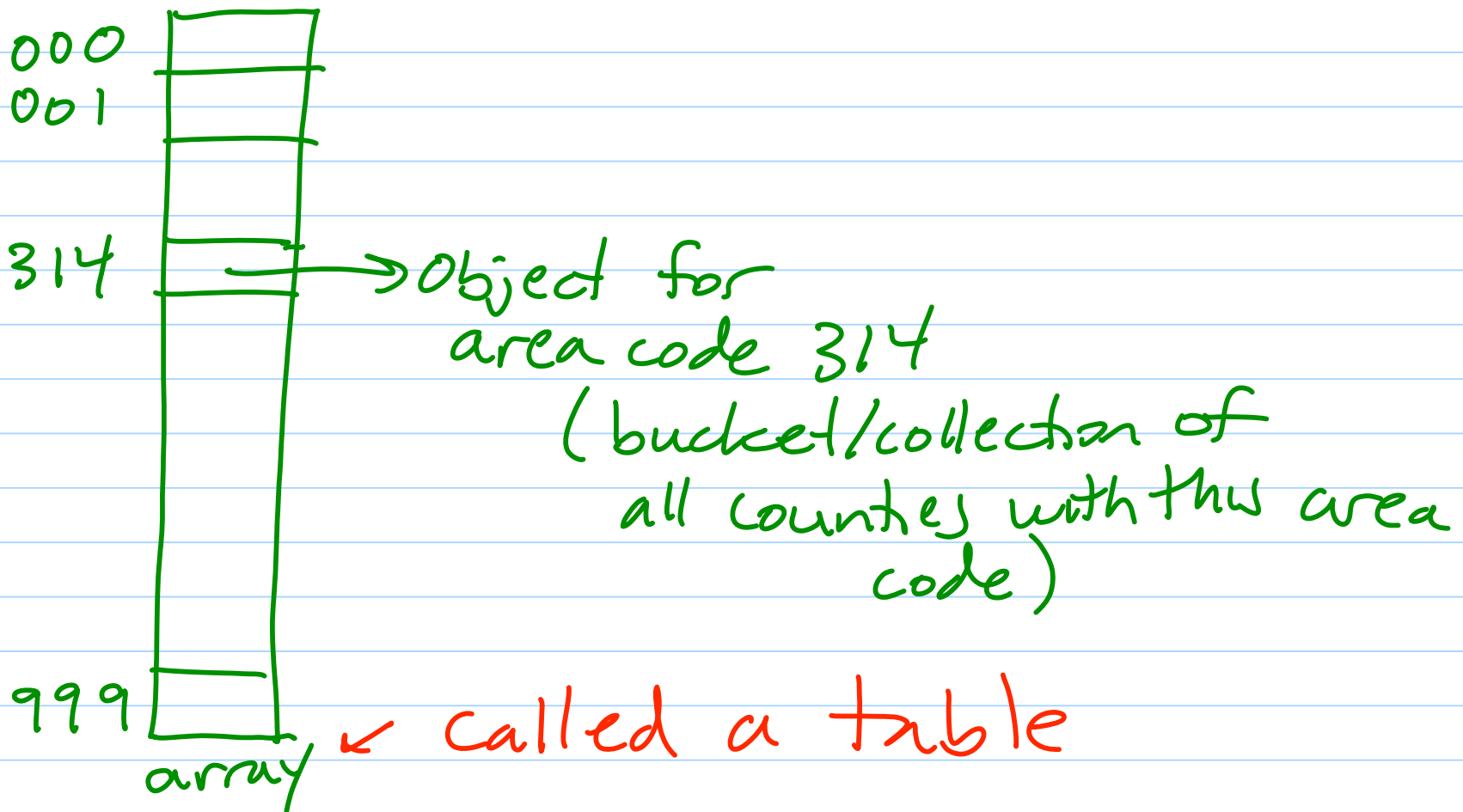
next class: Separate Chaining

Suppose we have a set of n area codes that we want to maintain

area code, for example, could be one instance var in a "County" object

Direct Addressing

insert, locate, remove



Key here

every element you might possibly
insert into set has a dedicated

index \rightarrow slot in the table

worst-case

insert - table[slot] = element $\Theta(1)$

locate - access table[slot] $\Theta(1)$

remove - table[slot] = null
EMPTY $\Theta(1)$

Let's make this a little more
general

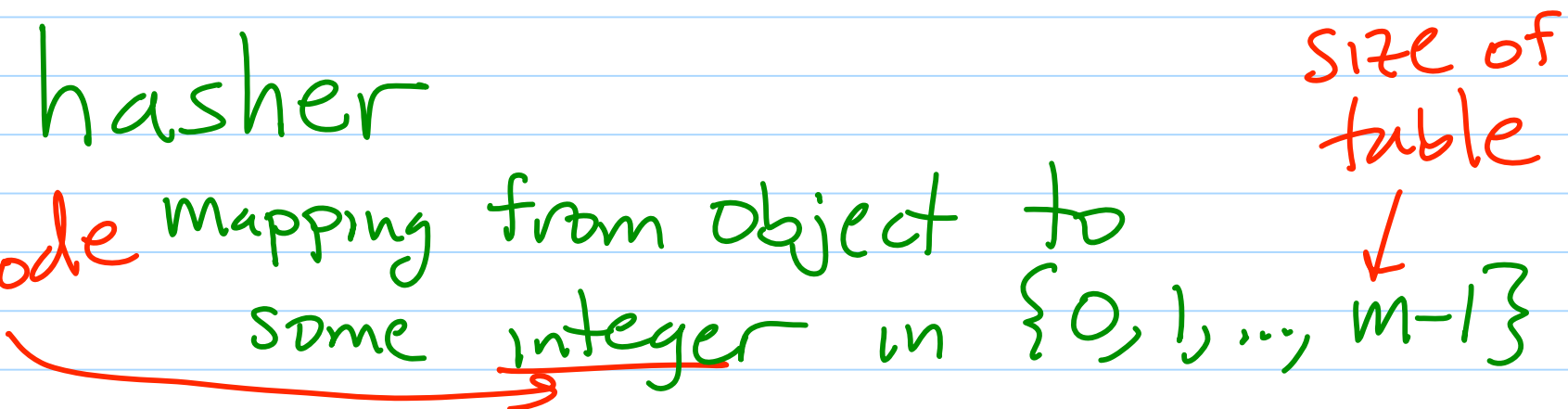
equivalence tester

define how check for equivalence

hasher

hashCode mapping from object to
some integer in $\{0, 1, \dots, m-1\}$

size of table
↓



What is the big limitation?

Size of table is as big as
 \cup { universe of all possible elements
that might be inserted

Consider a univ of 5000 students
where SS # is the id #.

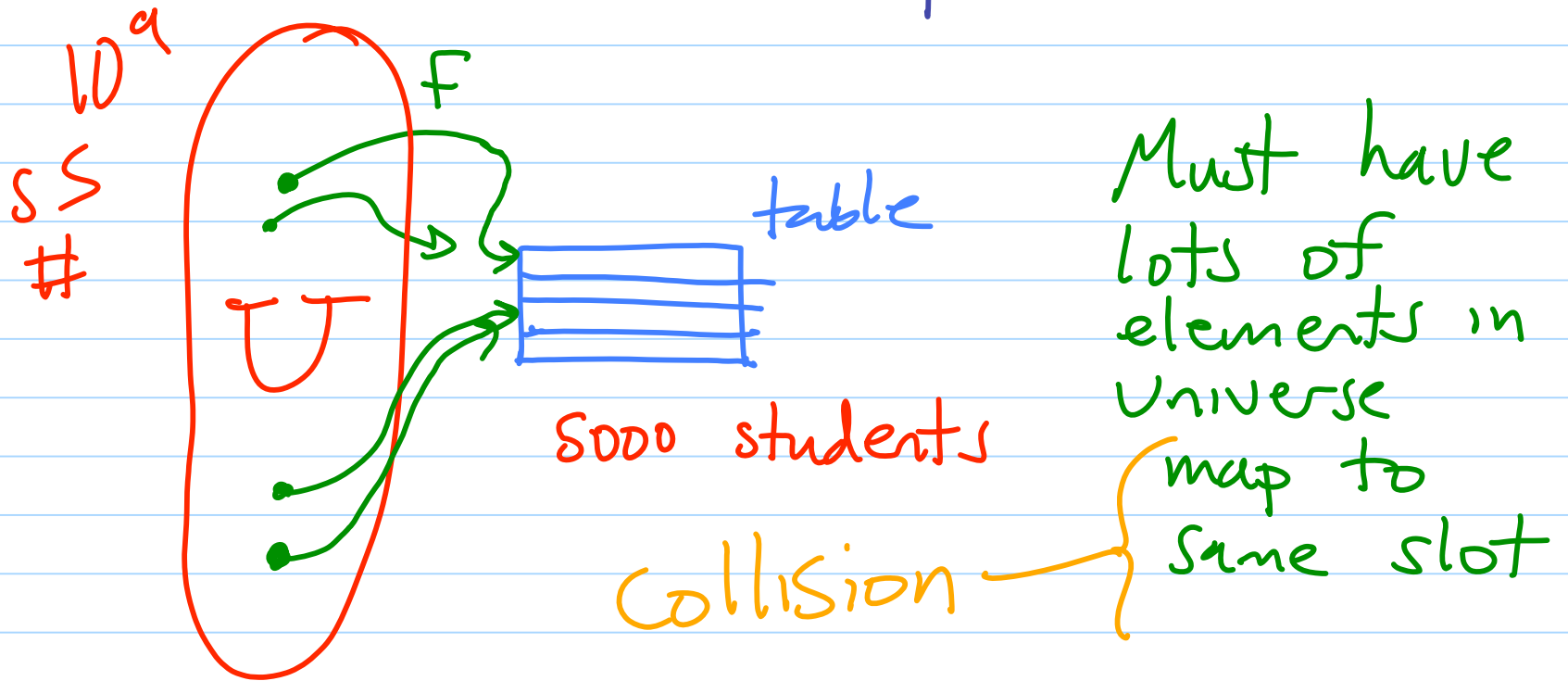
10^9

Direct addressing is only a reasonable choice (in terms of space usage) when

$$\text{roughly } n > \frac{|U|}{4}$$

↑
elements
held in set

We want time efficiency of direct addressing but we can't waste so much space



Hash Function

function that maps from hashcode

to $\{0, \dots, m-1\}$

object \rightarrow int

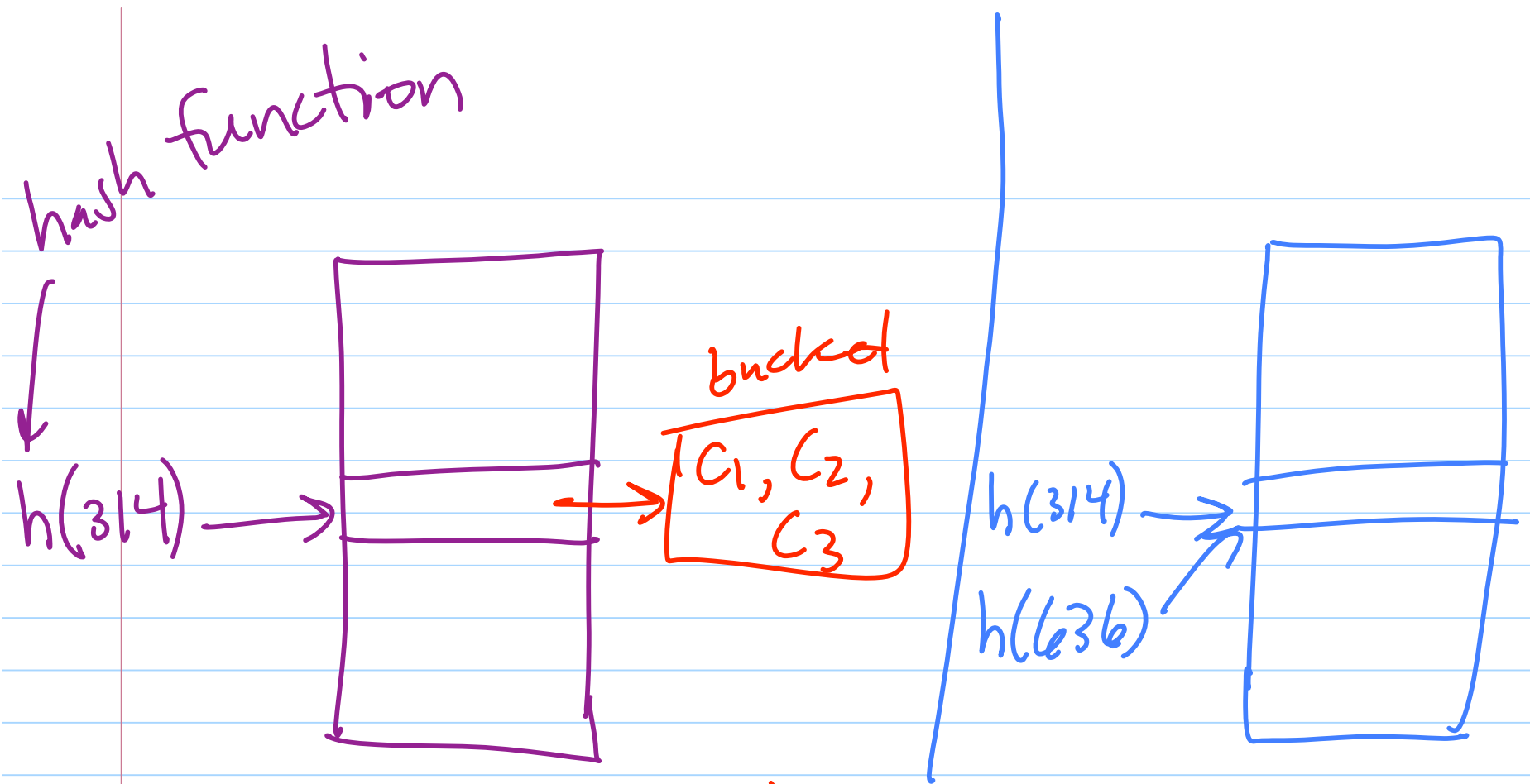
some
integer
from $0, \dots, m-1$

hash table size

Desired property — for each element $x \in U$, $\text{hash}(x, \text{hashcode}())$ is equally likely to be any int in $\{0, \dots, m-1\}$ prob $1/m$

Still we can have collisions -
what do we do?

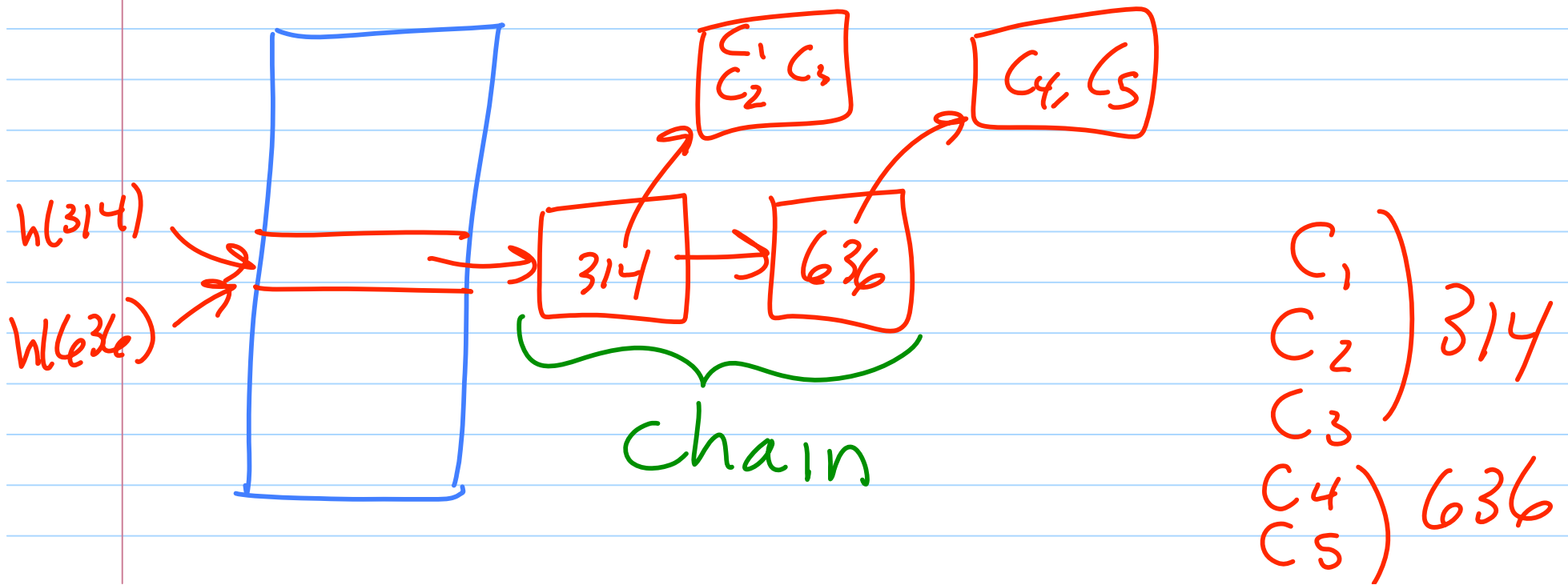
Open Addressing } these data
Separate Chaining } structures
} differ in
} how collisions
} are
} handled



C_1 C_3) area code) Bucket
 C_2 C_3) 314) Mapping

Separate Chaining

One solution: Keep a list of all elements (buckets) that hash to the same slot



Open Addressing

If the slot you hash to is already occupied (there's a collision), go somewhere else.

Requires there's \geq one slot per element. $(m \geq n)$

How do we decide where to go next?

Important that element e always follows the same sequence of slots as it looks for an open one.

probe sequence